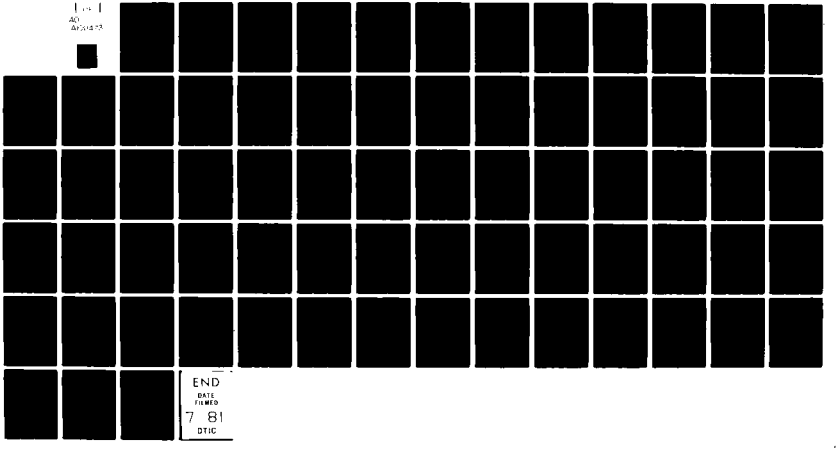
AD-A100 473 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
COMBINED QUARTERLY TECHNICAL REPORT NUMBER 21, SATNET DEVELOPME--ETC(U)
MAY 81 R D BRESSLER MDA903-80-C-0353
UNCLASSIFIED BBN-4679 NL



END
DATE FILMED
7 81
DTIC

Bolt Beranek and Newman Inc.

Report No. 4679

Combined Quarterly Technical Report No. 21

SATNET Development and Operation
Pluribus Satellite IMP Development
Remote Site Maintenance
Internet Development
Mobile Access Terminal Network
TCP for the HP3000
TCP-TAC
TCP for VAX-UNIX

May 1981

Prepared for:
Defense Advanced Research Projects Agency

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A100473 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMBINED QUARTERLY TECHNICAL REPORT No. 21	5. TYPE OF REPORT & PERIOD COVERED 2/1/81 to 4/30/81
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. D. Bressler	8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0353 & 0214, N00039-78-C-0405, N00039-79-C-0386, N00039-80-C-0664, N00039-81-C-0408
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order Nos. 3214 and 3175.17
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	12. REPORT DATE May 1981
	13. NUMBER OF PAGES 64
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DSSW Rm. 1D, The Pentagon Washington, DC 20310 NAVALEX Washington, DC 20360	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE/DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer networks, packets, packet broadcast, satellite communication, gateways, Transmission Control Program, UNIX, Pluribus Satellite IMP, Remote Site Module, Remote Site Maintenance, shipboard communications, Terminal Access Controller, VAX.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Quarterly Technical Report describes work on the development of and experimentation with packet broadcast by satellite; on development of Pluribus Satellite IMPs; on a study of the technology of Remote Site Maintenance; on the development of Inter-network monitoring; on shipboard satellite communications; and on the development of Transmission control protocols for the HP3000, TAC, and VAX-UNIX.

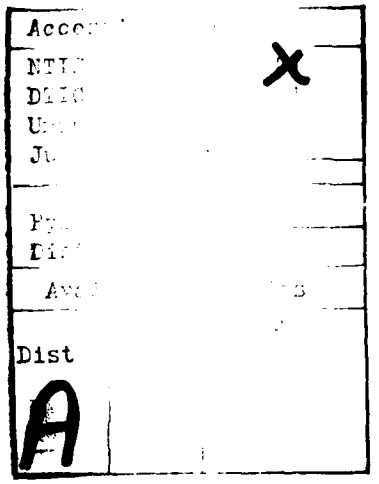
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4679 Bolt Beranek and Newman Inc.

COMBINED QUARTERLY TECHNICAL REPORT NO. 21

SATNET DEVELOPMENT AND OPERATION
PLURIBUS SATELLITE IMP DEVELOPMENT
REMOTE SITE MAINTENANCE
INTERNET DEVELOPMENT
MOBILE ACCESS TERMINAL NETWORK
TCP FOR THE HP3000
TCP-TAC
TCP FOR VAX-UNIX



May 1981

This research was supported by the Defense Advanced Research Projects Agency under the following contracts:

N00039-78-C-0405, ARPA Order No. 3175.17
N00039-79-C-0386
MDA903-80-C-0353, ARPA Order No. 3214
MDA903-80-C-0214, ARPA Order No. 3214
N00039-80-C-0664
N00038-81-C-0408

Submitted to:

Director
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Attention: Program Management

Table of Contents

TABLES

1 INTRODUCTION

This Quarterly Technical Report is the current edition in a series of reports which describe the work being performed at BBN in fulfillment of several ARPA work statements. This QTR covers work on several ARPA-sponsored projects including (1) development and operation of the SATNET satellite network; (2) development of the Pluribus Satellite IMP; (3) Remote Site Maintenance activities; (4) inter-network monitoring; (5) development of the Mobile Access Terminal Network; (6) TCP for the HP3000; (7) TCP-TAC; and (8) TCP for the VAX-UNIX. This work is described in this single Quarterly Technical Report with the permission of the Defense Advanced Research Projects Agency. Some of this work is a continuation of efforts previously reported on under contracts DAHC15-69-C-0179, F08606-73-C-0027, F08606-75-C-0032, MDA903-76-C-0213, and MDA903-76-C-0252.

2 SATNET DEVELOPMENT AND OPERATION

2.1 Automatic Stream Service for Gateway Traffic

As an adjunct to normal datagram service, we designed and implemented a facility which will automatically create and maintain low-capacity streams for gateway traffic. By providing one-hop satellite service and by aggregating several small messages into a single stream packet, the interactive user should see improved performance. Messages that would overflow space in the stream reservation, however, are sent as datagrams rather than discarded. This facility was implemented because of apprehension regarding the quality of service to be expected for interactive traffic from Europe when SATNET ceases to support direct ARPANET connectivity to London.

The new facility can be best understood by describing its two cooperating halves. One module creates and maintains a dedicated stream for each of the Satellite IMP's external hosts. Current default stream parameters specify a data size of 66 words and a reservation interval of 320 milliseconds (the default duration of a PODA frame), although these values may be overridden using the TENEX program EXPAK. Since each TCP message requires a minimum of 28 words of overhead (8 words SATNET, 10 words Internet, and 10 words TCP), the stream data size allows room for aggregating two small TCP packets.

The second module decides which messages should be sent using the stream. In order to prevent bulk data transfers from consuming all the stream bandwidth, a test is applied to messages arriving from the host to select those messages deserving the lower delay. Qualifying messages are added to the stream queue until the stream capacity is reached; additional messages arriving before the onset of the next stream interval are sent in datagram mode, as are messages larger than the stream allocation.

The implementation includes the flexibility to allow user evaluation of different selection criteria. Three parameters, which may be manipulated using EXPAK, determine the selection process. These parameters specify: (1) the SATNET "chunk buffer" index for selecting a key word in the SATNET or the Internet header; (2) a mask to apply to the key; and (3) a value the masked key must equal in order for the message to qualify for stream service. With this facility, messages can be selected based on such fields as the SATNET type of service, the Internet type of service, or the Internet protocol number.

The key can reference any one of 7 words of internal information, 3 words of SATNET header, or 10 words of Internet header. Internet options and the entire TCP header are not accessible. Particularly significant key words are the SATNET message control word at offset 7, the Internet type-of-service word at offset 10, and the Internet protocol number at offset 14.

During initial testing, the parameters will be set to accept all TCP messages regardless of type of service.

Maintaining the stream requires cooperation between the stream maintenance module and the acceptance decision module. When the acceptance decision module is unable to find a stream for sending messages, it sets a flag indicating that qualifying traffic is present with no stream. When the stream maintenance module sees this flag, it creates a new stream according to the predetermined parameters and informs the acceptance decision module that the new stream is available. Old streams can disappear in one of two ways. First, to prevent wasted bandwidth, SATNET will delete any stream not used during a one-minute time-out period. Second, when new stream parameters are specified, the old stream is deleted, so that the new parameters can take immediate effect.

To avoid including the stream ID in each datagram, we expanded the stream packet header to include the stream ID number. Thus, acceptable datagram messages are added directly to the stream queue without alteration. This number must be sent with each stream packet so that any site may determine when the stream was last used. Previously, the first data message in the stream packet always contained the stream ID.

The following table summarizes the new EXPAK-controlled

parameters inserted into the Satellite IMP for controlling the channel bandwidth assigned to the streams. Among these parameters is the SATNET stream repetition interval, specified as a 26-bit number in units of 10 microseconds; the "Stream Interval Low" parameter contains the low order 16 bits, while the "Stream Interval High" parameter contains the high order 10 bits. Host 1 refers to the DEC PDP/11-40 gateway attached to each Satellite IMP, while host 2 refers to the Internal Gateway module.

EXPAK PARAMETER	SIGNIFICANCE
Host 1 parameters	
662	Host 1 Stream Data Size
663	Host 1 Stream Interval High
664	Host 1 Stream Interval Low
Host 2 Parameters	
665	Host 2 Stream Data Size
666	Host 2 Stream Interval High
667	Host 2 Stream Interval Low
Message Selection parameters	
670	Chunk buffer index of key
671	Mask to apply to key
672	Value masked key must equal

It is expected that the automatic stream service will cause a larger number of packets to be received out of order due to the different delays between datagram and stream messages. The end-to-end user protocol, such as TCP, must reestablish packet ordering.

2.2 Modification of 1822 Host-to-IMP Protocol Module

The 1822 Host-to-IMP protocol module in SATNET was modified for simplifying system operation. In particular, we removed support for the old 32-bit ARPANET leaders, standardized the use of the 1822 "link" field, and modified the Internal Gateway to use the Internet byte count. (The Internal Gateway module, which allows the Satellite IMP to operate as a host on another network's IMP, performs the necessary format translations for messages to pass between the two networks and, based on a static routing table, redirects incoming or outgoing messages towards their eventual destinations.)

The SATNET 1822 protocol module allows hosts implementing the 1822 Host-to-IMP protocol with 96-bit headers to gain access to SATNET without implementing the special-purpose Host-SATNET protocol. SATNET-specific functions, such as stream service or priority and delay classes, are not fully supported, though. In addition, no acceptance or rejection messages (e.g. RFNMs) are implemented, so reliable transmission requires end-to-end error control. Hosts using the 1822 protocol and hosts using the Host-SATNET protocol can coexist without difficulty. In spite of the fact that the interpretation of the 8-bit "link" field differs in the two protocols, a source host using one protocol can send to a destination host using either protocol without knowing *a priori* which protocol the destination host is using.

When delivering messages to the Satellite IMP, hosts must supply the SATNET address of the destination in the "IMP" field of the 1822 header and must zero the "Host" field. In messages received from the Satellite IMP, the SATNET address of the source will be in the "IMP" field, and the "host" field will be zeroed. This addressing convention is consistent with existing rules for creating an 1822 header from the corresponding Internet address.

Both the 1822 header and the Host-SATNET header include an 8-bit "link" field used for host-to-host communication. With the exception that Internet messages from an 1822 host must have link values in the range 155-158 to designate Internet traffic, the source host may specify the link arbitrarily. Thus, non-Internet hosts, if any existed on SATNET, could use this field in the implementation of a local-net host-to-host protocol.

2.3 Software Problems Fixed

In the process of checking the 1822 Host-to-IMP protocol module, we discovered a problem in which maximum-size SATNET messages were being rejected by the Internal Gateway module in the Satellite IMPs. The cause was that the number of VDH words received from ARPANET is an inflated total because an extra bit, appended by the source host's 1822 Host-to-IMP interface, is counted as an extra 16-bit data word. To solve this problem, we

modified the Satellite IMP internal gateway to use the Internet Protocol "byte-count" field instead of the number of VDH words received from ARPANET for calculating the length of received messages.

We also found a bug that let slightly oversized packets infiltrate into SATNET, where they caused internal problems far removed from the point of entry. Once this was fixed, problems in passing TCP packets through the Satellite IMP Internal Gateway appeared. This was traced to another bug in which the conversion of the Internet header byte count to a word count employed truncation instead of the integer upper bound. Odd length packets lost their last byte and were then rejected by the gateway because they were too short for what the Internet header said. After fixing this problem, TCP packets were passed correctly.

A capability was added to the Echo Fake Host whereby it will emulate a TCP or UDP port swapper. Specifically, if a packet received by the Echo Fake Host is a TCP or UDP packet, and the destination port number is 7 (the assigned echo port), then the Echo Fake Host will swap the source and destination port numbers before returning the message. If the destination port number is 9 (the discard port), the message will be discarded.

For user convenience in diagnosing channel problems, several

variables within the Satellite IMP which control and monitor the PSP terminal were moved to page zero. The memory locations of these variables will now remain invariant with future Satellite IMP releases. The variables and their new locations are:

```
161/  TMWORD      ;command word for CMM module
162/  TMANS       ;answer received from CMM module
163/  ISTXMT      ;counter of packets transmitted
164/  ISTRCV      ;counter of packets rcvd OK
165/  ISTERR      ;counter of packets rcvd in error
```

We added a number of conditional assembly switches to the Satellite IMP software to control the assembly of different possible software configurations. One switch controls assembly of MATNET specific code, allowing most of the Satellite IMP source files to be shared between the two networks. A second switch controls memory assignments specific to a C/30 Satellite IMP, where a full 64K words of memory is available. A third switch causes inclusion of software support for an 1822 Host-to-IMP interface.

2.4 Hardware Problems Fixed

In response to a request from the NCC during a particularly lengthy outage of the military-controlled circuit between the SDAC IMP and the NORSAR TIP, SATNET was reconfigured to provide an ARPANET trunking circuit between the specified sites. In this

way, the NCC could examine the circuit from the remote end. Since only one ARPANET trunking path can be supported by SATNET at any given time, the London TIP was down for the duration that the NORSAR TIP was up.

Several more occasions occurred of the intermittent one-way line syndrome between the Goonhilly Satellite IMP and the London TIP. As in the past, the procedure for restoring service requires onsite personnel to loop momentarily the 9.6 Kbps Codex modem at either Goonhilly or London.

Goonhilly Satellite IMP operation failed, thereby isolating the London TIP, due to what was eventually diagnosed as a faulty frequency generator in the SPADE modem at Goonhilly. Once the British Post Office replaced the frequency generator, we could not restore site operation until after we performed a software reset of the Goonhilly PSP terminal.

A malfunction in the circuit between the Clarksburg Satellite IMP and the COMSAT Gateway was traced to a fault in the modem eliminator. We switched to the alternate channel in the modem eliminator to restore service.

Scheduled work on the main power transformer into the BBN facilities resulted in a power outage encompassing all BBN TENEX/TOPS-20 machines. Although SATNET monitoring continued to be available from directory SATIMP on ISIE, our TENEX utility

program for loading and patching Satellite IMPs was unavailable. Fortunately, no interruption to SATNET operation occurred, even though we were prepared to load failed Satellite IMPs from other Satellite IMPs with use of onsite assistance. Included in the list of machines with outages was the BBN gateway, which is the primary monitoring path into SATNET. Had the military controlled circuit between the SDAC IMP and the NORSAR TIP been down for the duration of the power outage, no SATNET monitoring path would have been available.

BBN maintenance personnel removed the 1822 Host-to-IMP interface from the Clarksburg Honeywell 316 Satellite IMP for delivery to the Mitre TIP, part of an ARPANET services rearrangement to provide COMSAT with a data link. This transition had no impact on SATNET services, since the interface was never enabled while at Clarksburg.

3 PLURIBUS SATELLITE IMP DEVELOPMENT

The major activities during the past quarter have focused on PSAT performance measurement and enhancement. In addition, BBN has supported the ongoing integration of the PSAT with other elements of the Wideband Packet Satellite Network. A third area worked on during the quarter is that of addressing in the Wideband Network. Each of these three general areas is discussed in more detail below.

Toward the end of February, the third PSAT was shipped from BBN and installed at the Defense Communication Engineering Center in Reston, Virginia. With three PSATs in the field, it was our intention to focus on multi-site system integration over the satellite channel. Unfortunately, the performance of the satellite channel (both in terms of error rate and stability) was inadequate to support such activities. PSAT/ESI channel tests were, therefore, deferred during most of the quarter while Western Union worked on the channel itself. During this period, integration of the PSAT and the miniconcentrator at Lincoln Laboratory proceeded. By the end of the quarter it was possible to demonstrate datagram transmission of speech from a source voice terminal on one LEXNET through the miniconcentrator, into the PSAT, and back out through the miniconcentrator to a second voice terminal on a different LEXNET. At the request of personnel at Lincoln Laboratory, one of the redundant host

interfaces on the Lincoln PSAT was split to provide support for two-host testing.

On February 10, we met with C. Weinstein and H. Heggestad from Lincoln Laboratory to discuss Wideband Network/EISN experiment planning. As a result of this and other discussions with Lincoln personnel, a potential system incompatibility in the Wideband Network was uncovered. In particular, the underlying PSAT frame rate of 21.2 milliseconds was slightly slower than the underlying frame rate of the voice terminal equipment (20 milliseconds). After a variety of interactions among BBN, Lincoln Laboratory, and Linkabit personnel, it was tentatively agreed that Lincoln would modify their speech parcel generation rate to be slightly slower than the PSAT frame rate rather than slightly faster.

The major activity during the quarter focused on analysis and incremental enhancement of PSAT performance. We began this activity with a review of all the key channel protocol module (CPM) tasks. The tasks within the PSAT fall into two major categories, processes and strips. Processes are software dispatched tasks which are scheduled on a round robin basis and require a greater run time overhead although they are easier to develop. Strips, on the other hand, are priority scheduled hardware dispatched tasks (based on the Pluribus PID device) and, while harder to write initially, have lower run time overhead.

The review of the key CPM modules has focused on (1) streamlining and combining various task functions, (2) replacing less efficient intertask communication mechanisms based on queues with more efficient specialized mechanisms, (3) eliminating unnecessary task activations, and (4) converting processes to strips. Host protocol module (HPM) task optimization will be the focus of subsequent work.

To provide a more detailed understanding of microscopic behavior of the PSAT software and to guide the software "tuning" activity, we have developed a process/strip measurement capability. This instrumentation allows us to measure the number of activations of a strip or process during an arbitrary interval as well as to determine the minimum, maximum and average run time for any task within the system. Although similar measurement capabilities have been developed for use with the Pluribus IMP, the capability that we developed for the PSAT is both easier to use and has less impact on the measured results than earlier measurement tools.

A third direction of our PSAT performance analysis activity involves the development of a PASCAL program (model) which given a semi-arbitrary traffic mix determines the number of Pluribus instructions per second required to support that load. This program, based on actual instruction counts, has been used both to guide the incremental software enhancements and as a means of

validating the process/strip measurement tools described above. Instruction counts can be mapped into module run times based on an estimate of the average instruction execution rate. Some measurements carried out during the quarter related to PSAT instruction execution rate are described in Section 3.1 below.

A performance enhancement activity described in several previous reports is the SuperSue Poller designed to eliminate the latency problem which the PSAT experiences at present when servicing high speed interfaces. During the quarter we completed writing the poller microcode for the augmented 2901 microengine using a modified version of the microassembler originally developed for the Voice Funnel project. This microassembler has greatly facilitated the development of the poller device as well providing a facility for additional SuperSue microcode enhancement which may be carried out in the future. The basic design of the SuperSue Poller is described in more detail in Section 3.2 below.

As noted above, the addressing within the Wideband Packet Satellite Network was another area that was given attention during the past quarter. Stimulated by W-Note 16 distributed by J. Pershing (BBN) at the November 1980 Wideband Meeting at ISI, a number of technical exchanges on the addressing issue have taken place over the last several months. Three approaches proposed by members of the PSAT group are contained in section

3.3 below. The intention of these proposals was to support flexible internet addressing within the Wideband Network while attempting not to disrupt the existing PAST addressing mechanism significantly. During a meeting at BBN on March 18, 1981 attended by R. Rettberg, J. Forgie, and R. Kahn, a consensus approach emerged. Subsequent to this meeting an analysis of the impact of this consensus addressing design on the existing PSAT implementation was carried out. Changes to the PSAT software associated with the new addressing scheme appear to be limited to the two host addressing tables PERHST and HOSTS and those modules of software which create and manipulate them (host address lookup, table initialization, group address creation, and the internal gateway).

3.1 PSAT Instruction Execution Rate Measurements

A set of experiments carried out during the quarter has provided information on actual average instruction execution times in the PSAT hardware/software environment. In these experiments a "typical" PSAT instruction (Load Register from Memory Indexed) was executed a large number of times under various conditions related to the placement of the instruction and its data in local or common memory. The results of these measurements are indicated below. The values are average

instruction execution time in microseconds.

	LOCAL INSTRUCTION	COMMON INSTRUCTION
LOCAL DATA	5.6	8.0
COMMON DATA	6.7	8.8

These numbers were all measured with one processor running the fake task of repeated "typical" instructions and the remaining processors executing the PSAT application. Additional measurements were made of the average instruction execution time for the "typical" instruction running out of local memory with local memory data and no contention (4.1 microseconds) and with contention from another processor on the same bus also running the same task of repeated "typical" instructions (4.9 microseconds). Based on our current understanding of the placement and execution of PSAT code, variables, and data, we expect that the average PSAT instruction actually takes approximately 7 microseconds to execute. In addition to providing a basis for mapping instruction counts into strip times, these measurements have also provided information for evaluating the impact on PSAT performance of various performance enhancement options such as expanding available local memory or rearrangement of code within the existing memory configuration.

3.2 SuperSue Poller Development

The SuperSue poller (SSP) is intended to solve the latency problem in servicing high data rate devices like the Satellite Modem Interface (SMI) and the High Speed Modem (HSM). These devices place severe demands on their service routines when they are transferring small packets. For packet lengths on the order of ten words (a typical minimum for these devices), the SMI requires service within about 50 microseconds for 3MBit/s operation (the HSM in about 100 microseconds for 1.5MBit/s operation), in order to handle the packet throughput rate. Since a Sue processor running in the Pluribus cannot service an interface with a worst case latency of less than about 300 microseconds (assuming it services both sides of a single device), a separate special piece of hardware is necessary to meet the demands of these devices.

This requirement led to the current approach of using a SuperSue processor card with a special set of microcode ROMs containing the poller function. Preliminary estimates indicate that the SuperSUE poller can handle two devices with a worst-case latency on the order of 50 microseconds, using a variation of the poller interface used in the current PSAT implementation. The SSP is designed to handle buffer-level control of two standard format DMA devices. The SSP handles the transmit and receive sides of each device. It communicates with the main

processors in the Pluribus by means of its internal registers, a series of tables in common memory, and through the PID.

The internal registers of the SSP are only accessible when the poller is halted (due to the use of the SuperSUE processor card). These registers are used to set up the addresses of the two devices to be serviced, information about the poller interface tables, and PID values to be poked when servicing of a device side has been completed. When the SSP is started (by writing into its control register), these registers are used to control the polling loop. If the SSP is subsequently stopped, the values in the registers are not guaranteed to be those written in originally.

The poller interface tables are a set of circular queues containing a number of interface service records. Each device has a pair of these queues, which must be contiguous in memory. In addition, both devices' tables must be adjacent on the same page in common memory. Each device table is composed of a receive queue followed by a transmit queue. The SSP permits all four queues to be of different sizes.

The queue entries are four words long, containing a status word, a word reserved for the user, a startpointer word, and an endpointer/flag word. The first word contains the receive status when the entry is a filled buffer. The endpointer references the

end of the buffer in the case of an empty buffer, or contains the value of the receive endpointer if the buffer is full. The least significant bit of the endpointer is used as a flag. If the SSP needs a new empty buffer and the next entry in the queue has a zero flag, the SSP will not be able to provide a new buffer to the receive side of the device until this entry is flagged as an empty buffer (flag bit = 1). Note that the SSP does not write the receive status register.

The transmit queue entries are identical in format to the receive queue entries, but are processed slightly differently. The startpointer is only read. The status and endpointer words are read when setting up the buffer for transmission and rewritten with the status and endpointer values at the completion of transmission. The flag is used to distinguish full from empty buffers. A non-zero flag bit indicates that the buffer has not yet been sent. The SSP will write the transmit status register of the device periodically with a value of 0 in order to inhibit the watchdog timer activity normally present on standard DMA devices.

The PID is poked with the appropriate number (kept in an SSP register) whenever servicing of a device side is completed. This is the end of transmission of a buffer on the transmit side of an interface, and completion of reception or the filling of a buffer on the receive side. Devices which are serviced by the SSP

should typically have their transmit and receive PIDs set to zero, since the SSP provides the necessary PID poking.

3.3 Three Wideband Network Addressing Proposals

The following paragraphs describe three alternative addressing schemes proposed for addressing within the Wideband Network.

Proposal #1 - "The Shared Field Approach"

All WB hosts can be addressed using the following internet address:

"28"
<16-bit PSAT logical address>
<8-bit local net/site address>

Although this looks identical to the W-Note 23 proposal, we add the following twist. To permit flexibility in expanding the size of local nets beyond 256 we would allow the PSATs to map several contiguous logical addresses to the same HAP process and physical port when a local net with more than 256 addresses is required. In a sense, the PSATs and local nets would share one or more bits of the internet address. In particular, if a local net wanted to be set up for 1024 address rather than 256, the local net would simply use the 10 low order bits of the 24-bit internet "rest" field and the PSAT would continue to use 16-bit

addresses with the low-order two bits "don't care" when the high order 14 match the address of the greedy local network in question. The PSAT will have 4 entries for the local network in its host table, all of which point to the same physical port. This approach supports local networks of variable size without restricting the PSAT logical addressing unnecessarily. Most local networks will probably be able to live forever with 256 local host addresses. Of those which cannot, some will be aware of this fact a priori and can initially be allocated multiple contiguous PSAT host addresses as required. For those which initially think 256 is enough but later find that they need more local addresses, additional contiguous address space can be allocated by the PSAT. Since this additional address space will not be contiguous with the previous address space in general, we would plan to reassign new internet addresses within the new address space to the original 256 hosts in the old address space (and perhaps reclaim the old space) at some point.

Proposal #2 - "The 12-bit PSAT Address Approach"

All WB hosts can be addressed using the following internet address:

"28"
<12-bit PSAT logical address>
"0000"
<8-bit local net/site address>

This is the approach initially proposed by R. Kahn with an additional 4 bits of PSAT addressing accommodated in the internet address. Although an address field that is not sized as an integer number of bytes is less than optimal for the PSAT, this is probably not a significant problem given that the PSAT's internal gateway is not expected to support a large amount of traffic. A requirement for more than 12 bits of PSAT logical addressing could still require that additional bits of the internet address be allocated in the future. We would prefer not to assign addresses in bit-reversed order to deal with this since we currently use the logical address as a direct index into a table and bit-reversed order would either mean reversal for all PSAT traffic or different internet and local addresses to get to the same PSAT host.

Proposal #3 - "Escape to 16-bit Addressing Approach"

All WB hosts can be addressed using the following internet address:

"28"
<8-bit PSAT logical address>
"00000000"
<8-bit local net/site address>

This approach again is basically the one proposed by R. Kahn. All 8-bit PSAT logical addresses except "zero" would correspond to directly connected or transparently multiplexed HAP

subscribers. The low order 8 bits are left for use by the individual sites in this case. To allow the PSAT to address more than 256 hosts, however, if the 8-bit PSAT logical address field is zero, we interpret this as an escape from the format above into one where the low order 16 bits of the internet "rest" field are interpreted as a full 16-bit logical PSAT address. PSAT hosts 0-255 could be addressed in either of two ways although one can agree that they are addressed without the escape feature. The escape would be used to address groups and internal PSAT hosts.

4 REMOTE SITE MAINTENANCE

4.1 CINCPACFLT Exercise

During the first two weeks of March, CINCPACFLT conducted an extensive wargaming exercise using the ACCAT system from the Oahu Remote Site Module. This exercise was an unusual test of the entire system, since it ran twenty-four hours per day for about ten days. Although the ACCAT Central site is up around the clock, operational staff and service are normally supplied during the business day only. Special procedures were adopted to assure that problems at the RSMs could be handled promptly. These included arrangements to have BBN staff members on call throughout the test. The keys were loaded and tested each day, so an "emergency RSM" could have taken place with little notice. It turned out that none was needed. The overall performance of the system was judged adequate, although the stresses placed on it were far beyond those anticipated in the original planning.

There were several hardware problems in the first few days; the exercise started at 2000 HST on a Sunday evening so the system had to be brought up from an absolutely cold start. There were a series of telephone converstations between the RSM and BBN staff members and the problem was identified as a memory failure. The continued off-on cycling of the system, which is now done daily, is not a recommended practice.

The Hawaii-West Coast satellite link was out for twelve hours. The singly-connected ARPANET node at Hawaii is always subject to such an outage. An operational system should have at least two paths between any two points.

Finally, users are frequently annoyed by "network glitches", as they call them. These look (to the user) like short network outages; they cause the connections between the RSM and the central site to disappear. The reconnection of the RSM is fairly complex; the users became quite expert at it, and reduced the time it took from several hours to under 30 minutes. This is not to minimize the discomfort which one experiences when a failure of this sort occurs, but to indicate that the frequency of reconnection is high enough that the users practiced the recovery procedures.

It was clearly important to track down the problem; the site staff were asked to call the NCC whenever this happened. The first hypothesis was that the line was going down for very short periods. It turned out, however, that the cause was congestion at the Ames IMP. This causes packets to be discarded when there is a lot of traffic. When an NCP packet is discarded, an incomplete transmission is returned, and the system closes the connection, since it has no way of knowing which message lost the packet. Unfortunately, this problem cannot be solved within the context of NCP, especially if one wishes to keep the number of

messages in flight to a maximum, as one must to make effective use of the satellite channel bandwidth. The network software in the RSMs should be upgraded to TCP, which can deal with such a loss and recover.

4.2 Info

The UNIX system has an extensive on-line manual system which allows any user to examine any page of the UNIX User's Manual. In addition, fundamental papers on system components, tutorials, and other documents are also stored on-line. There are several problems which this on-line manual system does not handle adequately. For example, the following questions are not easily answered through the ordinary manual system.

Where do I find the tutorial on the editor?

What is the configuration of this system?

What is the assignment of terminal lines?

Furthermore, there is no convenient place to store reference cards for the more important commands, logs of system updates (such as the changes in the on-line manual), policy and procedure statements, or general advice.

Two relatively simple commands, called 'help' and 'guide', were developed to meet these needs. A user could ask for information about a specific topic, and a file would be printed out which described that topic. This approach has several important limitations.

The separation into 'help' and 'guide', while it seems fairly clear on the surface, is probably useless.

Many 'help' files were structured lists of information, automatically maintained, and although necessary, were not especially enlightening.

Since there was no easy way to lead users from one file to another, there was a tendency to write rather long and involved files.

'Info' is a further experiment to help users find necessary information. The first intention of this command is to allow relatively new and infrequent users to find out what they need to know simply. It should also provide a place for system consultants to record both questions and answers. Finally, it should be possible to get any piece of information recorded in the system, and to try almost any command, without leaving the 'info' environment.

The initial inspiration for this command was a brief description

of the similarly named command in a TOPS-20 environment. The current 'info' uses the UNIX file system to contain the information more naturally, and takes advantage of this system to model an information network rather than a simple tree.

'Info' uses small files describing various elements of the system, organized in a hierarchical file structure. At any given time, several of these files, those in the current directory, are easily accessible. This collection of files accessible in a location is called an "information group" in the description below; each of the elements in an information group is called an "information block".

The main directory is /usr/info, and when the command starts up, the program transfers to this directory by default, but the user may select any of the information subdirectories, if the name of that information group is known. If 'info' is invoked in verbose mode, it tells the user how to find out what 'info' commands are, along with the top-level list of information blocks. After this, the system prints a prompt (INFO:) and waits for input. The user may type one of the commands, or one of the names specified in single quote-marks in the list of information blocks.

The current commands within info are:

info prints the command list again;

?	also prints the command list again;
show	prints out an information block;
start	goes back to the starting point of the information system;
change	goes to an information group named in the command;
list	lists the names of the current information group members;
last	goes to the previous information group;
try	executes the command (with arguments) appearing after 'try';
quiet	don't automatically print the list when transferring to a new information group;
verbose	do automatically print the list;
location	tell the name of the current information group; and
quit	leave 'info' and return to the shell.

There are two types of names in an information group. They are not distinguished in the list, but do cause different things

to happen. If the user types one of the first kind, the referenced file is displayed. If the user types one of the second kind, 'info' moves to another directory; since this need not be a subdirectory of the current directory, or its parent, the data is effectively organized in a network rather than a simple hierarchy.

The user may get the name of the current information group at any time. This allows the user to interrupt a terminal session and return to the same place in the information hierarchy at a later time. For example, if the user were in the portion which described the manual page formating rules, the following dialog would get the user out of info and back in:

```
INFO:location
manual/format_rules
INFO:quit
$ info -q manual/format_rules
INFO:location
manual/format_rules
```

Where 'quit' returns the user to UNIX, and the user has specified the actual place to go on the command line when 'info' is invoked again. Quiet mode has been requested, rather than the verbose mode option in conversation with the system.

Any command may be executed during an info session, simply by typing 'try' before the command and its arguments. This means that a user can immediately try various options before leaving the information tree.

Persons who wish to create new portions of the information tree need to know where the information is stored and how to add new topics. All of the data is stored in the /usr/info hierarchy. A list of the pathnames in the initial data base is given in Table 1.

The brief description of the commands themselves is found in /usr/info/.infodata. The list command displays the file .infolist in the current directory, and a local information block name corresponds to the file name in the current directory. A name which causes a transfer is the name of a new directory, and the transfer causes the command to transfer to that directory. The relative pathname from /usr/info should be placed in the file .infodir. Since it is unnecessary for this directory to be either above or below the current directory, the information tree is transformed into a network.

The evolution of the data base can procede naturally if a system consultant always provides two answers for each question. The user should get a direct answer, and an information block should be entered into the system. There are a number of other

```
.infodata
.infodir
.infolist
commands
configuration
documents
new_user_info
news
procedures
use_info

commands/.infodir
         .infolist
         comm.list
         more.help
         msg.help
         support
         support_desc

configuration/.infolist
              disk
              lines

procedures/.infodir
           .infolist
           install.form
           install.req
           motd.form
           sys.install
           unix.names

documents/.infodir
          .infodirs
          .infolist
          doc_locations
          manix
          manual
          manual_rules
          prman

documents/manual_rules/man.content
                       man.format
                       man.macros
                       man.names

new_user_info/.infodir
              .infolist
              c_programming
              editors
              formatter
              printer
              tutorials

news/.infolist
     send_news
```

Table 1. Initial Info Data Base

improvements which are underway. In the latest version, which will be installed shortly, the UNIX User's Manual is accessed when appropriate. Experiments with an index are planned.

The entire 'info' command is implemented as a shell file. This permitted rapid implementation and extensive experimentation with features. The drawback to a shell file implementation is that the resulting command is perhaps a bit slower than one would wish. In the long run, the program should be converted into a faster form; for the time being, the flexibility of this implementation is still quite important.

4.3 *Version 7 Utilities*

The first phase of preparations for installation of Version 7 utilities at the RSMs is now nearly complete. The kernel installed at the sites (sys.130) can support the Version 7 concept of environment passing. A library which simulates the various Version 7 system calls has been prepared, and each command which is a candidate for distribution is tested in the simulated environment. The on-line version of the UNIX User's Manual has been modified to reflect the changes. Distribution of these commands will begin during the next quarter.

A new distribution file system will be constructed on the BBN system clear packs. This system will be thoroughly tested to

insure that it is complete; in particular that there is a Makefile or Build.info for each installed command and library, and that the libraries are consistent. This testing can be carried out during normal operations on the open packs, using the 'chroot' command which permits a high degree of isolation between the release test area and the rest of the disk. After these tests are complete, the release file system is transferred by tape to the secure packs, where a file test of completeness and consistency is made. Any decrepancies are hand transcribed to the open packs and the cycle repeated.

Certain important system utilities, including the shell, the C compiler, the library and 'make' will be transferred to the sites over the network in the next phase. The complete schedule for installation is not yet settled, but there will be a new kernel installed during the summer, and all of the new software will be installed by that time.

5 INTERNET DEVELOPMENT

Internet activities during the quarter were in five areas:

- o Operation and Maintenance of Gateways
- o VAN Gateway Development
- o Measurements and Investigation of the Internet
- o Planning for Transfer of Responsibility for Gateway Development
- o Investigation of Architectural Issues.

These activities were coordinated with two separately funded efforts concerned with the ARPANET Routing Study and performance testing sponsored by DCA.

5.1 Operations and Maintenance

Operations and Maintenance activities continued at a low level during the quarter. The installed hardware and software has been fairly stable during this period, and no unusual problems were encountered. This situation enabled us to direct most of our efforts into the research and developmental tasks which follow.

5.2 VAN Gateway Development

Acceptance testing of the VAN gateway implementation continued during the quarter. The VAN gateway is an amalgamation of software and hardware supplied by five separate groups. RSRE supplied the link level hardware and software interface. UCL supplied the X.25 packet level software. The ARPA standard gateway software was obtained from the Information Sciences Division of BBN. The MOS operating system was supplied by SRI. Finally, the VAN gateway-specific code, to provide the "glue" which binds all of the pieces into a system, was developed under this contract.

Efforts during the quarter concentrated on packet level acceptance tests, using the Telenet test facility which is accessed by a dial-up connection. This work primarily involved detecting and isolating bugs in the various software components mentioned above. Many of these problems are a result of the imprecision of the X.25 as a specification. We have gone through several interactions with RSRE and UCL personnel, to report problems identified during the tests with the Telenet facility. The software which RSRE and UCL provided had been fully tested and was working in their local X.25 environment; the problems which we encountered resulted from a difference between the European and Telenet interpretations of X.25, in particular in those areas where the specification is ambiguous or silent about

some detail. For example, RSRE personnel observed that Telenet's requirements may be at odds with the CCITT official specification as far as generation of frame rejects is concerned.

We have been modifying the various software components as necessary to meet requirements for Telenet approval. (As of this writing, approval was granted to connect to Telenet, although this was after the quarterly period covered by the present report.)

One major outcome of the testing efforts has been the realization that the technology of testing protocols is not very advanced. In particular, we note that the configuration we were testing is structured such that the testing proceeds smoothly only when there are no flaws in the system under test. Using the test facility to essentially debug the integrated implementation proved unwieldly and time-consuming.

One example of this is in the level two interface provided by RSRE. The implementation of this interface as an LSI-11 device is such that resetting the processor, which resets the I/O devices, hangs up the modem telephone connection. Since this is a common occurrence during debugging sessions, considerable re-dialing was necessary. Since the level two implementation was already tested at RSRE, most of this debugging should have been unnecessary; the main source of problems was the different

interpretations of the X.25 specification.

Similar problems were detected and isolated in the packet level code provided by UCL. After discussions with the UCL personnel, we concluded that the problems resulted from similar ambiguities, or from details of the protocol which were not needed in the environment for which the software was originally constructed.

Deployment plans for the VAN gateways are as yet undetermined, pending a decision on how and where to obtain a line accessing Telenet. Efforts during the coming quarter will attempt to do some simple tests in an internet environment, using an echo facility to be provided by UCL. We are investigating arrangements for non-interfering usage of a Telenet port already at BBN in support of another government contract.

5.3 Internet Experiments

In conjunction with efforts under the ARPANET Routing Study, we performed several informal experiments and fault-isolation investigations during the quarter. The primary goal of this work is to feed experiences with the current Catenet into the Study activities which are aimed at investigating architectural issues. One aspect of these issues is the question of monitoring the internet components, to develop and integrate the philosophy of

the CMCC, which is loosely coupled to the gateways it monitors, and the NU system, which is very tightly coupled to the IMPs which it monitors. We have suspended further work on the CMCC/NU integration, since there is currently insufficient space in the gateways to implement any additional monitoring facilities. Current plans are to resume this work in FY 82, in conjunction with transfer of gateways to more powerful machines.

One of the current anomalies which we have discovered is summarized below. It represents a problem of a class which is numerous in any large distributed system, and motivates the current need for more elaborate monitoring and fault detection facilities. As the system becomes increasingly large, it becomes virtually certain that, at any given time, there will exist some malfunction in the system. The system architecture and implementation must be directed toward an environment in which there is always something wrong, and include the embedded mechanisms to isolate and repair problems.

The anomaly is as follows. About once a day, the UCL gateway sends a trap saying "NDRE/4 up". The UCL gateway should not ever get anything from the NDRE gateway, because the Tanum Satellite IMP contains a patch to throw away specifically any traffic coming from NDRE and destined for UCL. This patch was put in place previously to eliminate the "Atlantic Shuffle" which has been discussed at several Internet meetings. However, UCL

knows that NDRE ought to be there, so it keeps sending it GGP messages. UCL will believe that NDRE is up if it gets any reply to these messages. Thus only one packet need get through to cause this event, not several within a short time. Every time we have seen the "up" trap, we have seen a "down" trap two minutes later. Two minutes is the timeout time for the current implementation.

Goonhilly's SATNET address is 74 (octal), and Etam's is 75, so they differ by only one bit. If the SATNET header on one of NDRE's GGP replies picked up that extra bit, then Tanum would forward the packet to Etam instead of Goonhilly, and the packet would be delivered to the BBN gateway instead of the UCL gateway (their host numbers are the same). If the Internet header was still intact, then it would still be addressed to UCL, so that BBN would send it back into SATNET, which would then deliver it to the UCL gateway. This is currently a hypothesis, which has not been proven, but is still under investigation. It is an anomaly in the system operation, which "can't happen", yet does. The CMCC was used to discover this situation.

5.4 Planning for Gateway Support

A major activity during the period involved developing plans for continued operations and maintenance of the current internet

system. There are two aspects of this activity. We were informed by DARPA during the quarter that full responsibility for operations and further development of the gateways would be transferred to this contract late in CY81. In addition, we were asked to determine a set of alternatives to provide the best possible service to the European users now served by the ARPANET service of the London and Norsar TIPs. This service is expected to be removed in CY81 with the decommission of the current satellite circuits which implement the inter-IMP communications links.

We held a meeting with personnel from the Information Sciences Division to review the status of the current efforts, and to develop a plan for transfer of responsibility by the end of CY81. We also developed a set of alternatives and recommendations for supplying service to the European users through the NDRE and UCL gateways.

NDRE had previously made arrangements to procure a commercial ring-type local network to use for their research activities. To provide service at NDRE in place of that provided by the Norsar-TIP, we plan to configure the NDRE gateway to interface SATNET and the ring network, instead of SATNET and the ARPANET as it does currently. Terminal access will be provided by a TIU, which NDRE will obtain from SRI directly.

Arrangements have been made, pending authorization, to procure two ring network PDP-11 interfaces (ProNet, based on the MIT V2LNI network) interfaces. These will be used to configure a small ring network at BBN, to develop the software device drivers to interface the gateway and TIU to the ring network. We will use the TIU and PDP-11/40 currently at BBN to support this work.

An alternative which was investigated involved using the Norsar TIP machine as a TAC, which supports TCP. This would require new hardware interfaces to be developed for the Honeywell 316, however. Since that machine is essentially obsolete, the TIU approach is preferable at this time.

The UCL configuration is similar. Here the higher usage of the London TIP cannot be handled by the TIU, however. Another goal is to have the full complement of equipment maintained by the NCC, to emulate ARPANET-style operations. The Cambridge ring network will be available to support local usage, but not in the needed time frame.

After discussions with the various parties involved, we are currently planning for a configuration which includes installation of a C/30 IMP, to implement a one-node clone of the ARPANET as a local network for UCL. The London TIP will be configured as a TAC and connected to the IMP. The gateway will then connect this new network with SATNET. Development of the

NCC monitoring and control facilities under a separate contract is in progress, to provide the capability of monitoring and controlling IMPs and TACs which are not on the ARPANET, by using the Internet Protocol.

We are also planning for the eventual replacement of the current PDP-11 gateway hardware with C/70 hardware, which will remove the address space limitations which severely limit the functionality and performance of the current gateways. Since this will take some time to effect, our efforts with the current system will concentrate on whatever changes we can make to improve performance and reliability for the short term. Such changes might include creating stripped-down versions of the current BCPL gateway, removing functions which are never exercised in the specific current configurations, or use of a gateway implementation done in assembly language in support of the BCR testing efforts at DCEC.

5.5 Architectural Investigations

Activities in this area were primarily in support of the efforts under the ARPANET Routing Study to investigate the Internet architecture. We have been working closely with the personnel involved in that study, to supply the experience from the current Internet activities, and help develop a uniform model

of the Internet as a system.

We expect to issue a series of IENs during the next quarter which present the issues of the Internet, and develop a plan for a phased installation of new gateway hardware, and new software based on this analysis work.

6 MOBILE ACCESS TERMINAL NETWORK

The major part of our participation in the development of the Mobile Access Terminal (MAT) and the MAT Satellite Network (MATNET) during the last quarter is the preliminary system integration being carried out at E-Systems, ECI Division, in St. Petersburg, Florida. Preliminary system integration includes interfacing the two BBN-built Red subsystems, which we shipped from BBN to ECI in the previous quarter, with two ECI-built Black subsystems. Communication tests are currently being conducted prior to acceptance testing by the Naval Electronic Systems Command (NAVELEX) in the Department of the Navy.

During this quarter, another major milestone in the project was passed; namely, we configured the two complete MATs at ECI as shipboard stations, enabling end-to-end user Teletype communications for the first time. To create two shipboard MATs, the gateway processor of the shore-based MAT was temporarily converted to a Terminal Interface Unit (TIU) processor by loading the appropriate software. In place of difficult-to-obtain satellite channel time, the ECI-built RF satellite channel simulator was used. This configuration was successfully demonstrated to participants at the MATNET status review meeting called by NAVELEX on February 12.

Although the end-to-end user Teletype demonstrations

revealed respectable system behavior when sending user data either word-at-a-time or line-at-a-time, immoderately long delays of the order of many seconds were experienced when sending single-character packets over the MATNET channel. The magnitude of the effect seen was unexpected, in spite of the fact that we recognized that low throughput for single-character packets should be expected due to the large overhead added to each packet independent of size. Potentially the major contributing cause for the long delays is a mismatched set of TCP parameters. We will address this problem after MATNET system integration is completed.

We found and corrected a buffer management problem in which a buffer was not being released correctly if a satellite channel I/O reset operation occurred after the creation of an output packet and before its transfer to the output routine. The problem first appeared at low signal-to-noise ratios, where the probability of issuing I/O resets was high due to increased difficulty in receiving control packets. In this situation, the MAT would enter a lockup state, wherein the Red processor would continually issue reset commands to the Black processor but would never finish the reset operation because of a lack of buffers.

System testing revealed another malfunction, whose symptoms were that the Red processor failed to achieve reservation synchronization because of a failure to receive control packets.

Further debugging revealed that PTI pulses from the Red processor failed to reach the Black processor due to a relay switch fault in the ON-143 COMSEC isolation box. As a temporary fix, we were able to restore operation to the ON-143 by momentarily switching power off; later the unit was replaced.

An intermittent cable fault at the board connector of an 1822 Host-to-IMP cable prevented the TIU from working properly with the Red processor. Compounding the difficulty in isolating the problem was that the replacement cable had an identical fault. Other hardware failures diagnosed and fixed include a C/30 I/O board, a C/30 memory board, and a COMSEC interface board in the Black processor. A receive PSK module in one of the AN/WSC-3 radios presented degraded performance until replaced.

We accepted formal delivery of C/30 packet switch processor #3, which is scheduled to remain at BBN for subsequent Red processor software development and problem resolution. Initially, this unit was installed into the MAT test-facility rack at BBN for acceptance testing. Having satisfactorily passed acceptance testing, the boards for C/30 #3 were shipped to ECI, where they were swapped with the boards for C/30 #1. Subsequently, the latter boards were shipped to the BBNCC repair facilities in Cambridge, Massachusetts, for retrofit of engineering changes. Afterwards boards from C/30 #2 will be returned to the BBNCC repair facilities in Cambridge for similar

retrofits. These changes, which are prerequisites to BBNCC maintaining the equipment, are scheduled so as to have two working MAT stations at ECI. For the duration of the C/30 upgrading process, we will not have a working C/30 at BBN dedicated to the MATNET project.

7 TCP FOR THE HP3000

7.1 Progress in General

Major progress was made in the HP3000 TCP project during the last quarter. Most of our effort has been concentrated on testing the existing code. So far we have been able to connect to other hosts, and log into them with our user TELNET. In addition, we have been able to connect to our TCP server TELNET with the VAX-UNIX user TELNET.

7.2 Problems in Particular

While we have had a great deal of success during the last quarter there have also been a number of problems. First, we have discovered difficulties with some of the HP systems software. In particular, both the terminal handler and the network interface handler seem to have some design flaws.

7.2.1 Terminal Handler Problems

The HP3000 terminal handler treats all terminals as if they were half duplex devices. While the terminal and controller hardware does not preclude full duplex operation, all of the system software between the hardware and user programs assumes half duplex operation. This design philosophy presents some

problems for interactive programs such as TELNET.

Normally, any interactive program written for the HP3000 assumes that all asynchronous input will come from the user terminal. The normal operating mode for these programs is to read from the terminal when user input is expected. Any data sent to the program via the terminal terminates the read operation while the program processes the data. The termination of the read operation prevents the user from entering any new data until a new read is initiated. This is normally done when the program expects more input from the user.

While this procedure works reasonably well when there is only one source on asynchronous input to the program, it will not work for a program such as TELNET which receives asynchronous input from both its network interface and a terminal. In this case the program has to be able to read data from two sources simultaneously. While HP provides a mechanism for handling this situation, it has some problems.

The mechanism provided by HP allows a program to initiate read operations on a number of I/O devices and wait for the first one to respond. In the case of TELNET, read operations are initiated for both the network interface and the user terminal. After initiating the read operations the TELNET process is suspended until it receives data. In theory the TELNET process

should be able to receive data, initiate another read operation and then process the data it just received.

Unfortunately this cannot work when TELNET receives network data from the foreign host. Most such data that TELNET receives is intended for the user and must be printed out on the user terminal. Since the terminal acts as a half duplex device, TELNET cannot accept input from the terminal while the write takes place. This forces the TELNET software adopt and interlock mechanism which terminates the terminal read operation, writes the network data to the terminal, and initiates another read operation. This operation creates a window during which any data *from the terminal* is lost.

Besides loosing any data the user was typing while TELNET writes to the terminal, there is a basic interlock problem between the write operation and the final read operation. If the write operation is not allowed to complete before the final read operation is initiated, some of the write data is lost. This acts to increase the amount of time after each TELNET write that the terminal is not available to the user.

Ostensibly, HP provides a system call which does not return until a write is complete. Unfortunately the call returns prematurely when a large amount of data is written to the terminal. This adds to the complexity of the read/write

operation interlock.

The correct solution to this problem requires a major revision of existing HP code. Unfortunately, the major revisions of the HP code are beyond the scope of this project for two reasons. First, there simply is not enough time budgeted in this project for major modifications to HP software. Any useful revision of the HP code would probably be an effort comparable to the entire TCP project. Second, any major revision to HP code will make it completely incompatible with the standard HP operating system. Any future revisions to the operating system distributed by HP would be almost impossible to integrate into the modified operating system.

7.2.2 Network Interface Problems

The network interface software consists of two modules, the MPE-IV INP (network interface controller) driver and the INP microcode. The problem we are experiencing occurs in the buffering scheme used by the driver. The driver code incorporates up to 7 send and 7 receive buffers. The driver copies data transmitted to and from the network into these buffers.

A problem occurs when there is a great deal of network traffic. Since the INP is a half duplex device it has a problem

handling a great deal of traffic in both directions. Data going out is sometimes delayed long enough to fill all seven output buffers. This causes subsequent network write calls to return an error condition. In order to prevent this problem a careful count must be kept of the number of INP buffers currently in use. While keeping such a count is possible, the operation requires additional overhead on each write operation.

A long-term solution to this problem would be to use two INP interfaces to the network. By dedicating one to network input and one to network output one could operate in full duplex mode and eliminate the bottleneck. Implementing this solution would require a major modification of the IMP code used to interface to the HP3000.

8 TCP-TAC

Major progress was made in the TAC project in the last quarter. All of the modules were assembled to run together, including the 1822 Host interface, Internet Protocol (IP), Transmission Control Protocol (TCP), Network Control Protocol (NCP), TELNET, Multi-line terminal controller (MLC), and site file configuration modules. The relevant code was debugged and the majority of it now works.

The TCP/IP portion of the TAC is now functioning. TELNET connections have been made to various types of hosts. These hosts include TOPS-20 (BBND,ISIE), TENEX (BBNC), Multics (MIT-Multics), Unix (BBN-Unix, EDN-Unix), VAX-Unix (BBN-VAX), HP (HP-3000), IBM (UCLA), and RT-11 (Comsat). Also, connections have been made to hosts on networks other than the ARPANET, including BBN-RCC, COMSAT, and BBN-LOCAL. Connections have also been made from one terminal to another terminal on the same TAC. The NCP portion of the TAC is still being debugged.

Currently memory usage in the TAC is looking adequate. This usage is broken down as follows:

Code	16,000 words
Trmblks	4,588
Buffers	12,180

The IP code takes 1060. words and the TCP takes 2744. words. These numbers are based on a TAC with 63 terminals in a 32K H-316. If the machine was a BBN C-30 with 64K, the extra 32K would go into buffers. It appears at present as if there are enough buffers for good terminal service.

Work remains to be done on the TAC in several areas. These are: reassembly of fragmented datagrams, urgent TCP data, stricter control of Message blocks, new hunting algorithm, and a host loader.

We are currently developing a protocol to monitor and control the TACs. This protocol will be used to collect status and statistics information and Trap reports. It is being designed to be used with the Internet protocol, which will allow supporting TACs on networks other than the ARPANET. Also, work is being done to enhance the Packet Core protocol to make it more reliable in an Internet environment.

The monitoring protocols will be implemented in the TAC and in the Unix-based Network Operational Center (NOC). In addition to being used in the TAC, these protocols will also be used in the IMPs. This will allow them to be supported on networks where there is no NOC. Like the TAC, this will make the IMPs supportable on networks other than the ARPANET. In particular, this will permit a C/30 IMP to be placed in London when Line 77

is no longer operational.

It is expected that the software will be ready to begin deploying TACs in early September.

9 TCP FOR VAX-UNIX

The past quarter's work on the VAX TCP Project included several major milestones. First, the implementation was released for general use within BBN on the BBN UNIX Cost Center's VAX. Second, the TCP was successfully installed at the University of California, Berkeley (UCB) Computer Science Research Group, our first beta test site. The TCP is now running on their research machine, and regular testing is going on between the two sites.

In addition to development and debugging work, we participated in a meeting of ARPA contractors that took place at UCB on April 6 and 7. The main purposes of the meeting were to discuss future interprocess communication (IPC) enhancements to the Berkeley kernel, and to discuss and provide guidance for future Berkeley system developments in general.

9.1 TCP System Work

9.1.1 System Debugging

Much of the last quarter was spent debugging the TCP. Several bugs were discovered and fixed which affected the system's stability and performance. These included a bug which was causing transmission beyond the edge of the foreign peer's advertised window, resulting in erratic and degraded performance

over time. Many of the protocol implementation bugs were discovered in debugging sessions with the TCP-TAC implementation also being done at BBN. As a result of the debugging effort, the system is now stable over long periods of use, and performance has been made consistent.

9.1.2 Adaptive Retransmission

In addition to the debugging work, some new development was done in the area of adding an adaptive retransmission policy to the implementation. The scheme is based on measuring the round-trip response of data packets being transmitted and acknowledgements being returned. When a data packet is transmitted, a timer is started. The timer is incremented once every second until all the data sent in the packet is acknowledged. The resulting time is added to a minimum offset and used as the retransmission time. In addition, each time a retransmission takes place, the current retransmission time is increased by a base value. Thus, if no response is received to a retransmission, the value will continue to increase until it reaches a maximum value. The round-trip timer is restarted for the next packet sent after it is read. Currently, no smoothing of the retransmission time takes place between successive round-trip timings, but this will be added if necessary.

9.1.3 Integration in New UNIX Kernel

The TCP implementation was done under an older version of the Berkeley kernel (3BSD) to avoid compounding the problems of developing significant kernel modifications with those of converting to a new, somewhat unfamiliar operating system. In preparation for the installation of the network software at UCB, the TCP was ported to the latest version of the kernel (4.1BSD). This version of the kernel runs significantly faster than 3BSD due to reduced system call overhead and use of more efficient algorithms in accessing system data structures. In addition, it provides better error reporting and operational characteristics, more general peripheral device handling, and increased user functionality.

A pre-release version of the 4.1BSD system was obtained from UCB and modified to include the TCP. This involved no major changes to the main body of TCP code. Minor changes were made in the way the network data structures are declared and accessed. The local network interface device driver was modified to work with the more generalized device handling scheme. This involved making use of some new system data structures pertaining to I/O devices, and generalizing the driver to handle multiple physical interfaces. This step will aid in the future addition of multiple network handling capabilities to the software (see below).

The integration of the code into 4.1BSD was successful and resulted in an observed 40% improvement in performance over the 3BSD version. This can probably be attributed to the reduced system call overhead. The 4.1BSD version is currently running at UCB. We will be converting the BBN UNIX Cost Center VAX to the newer version of the kernel in the near future.

9.2 Higher Level Protocol Software

Work has proceeded during the quarter in bringing up the higher level protocols: TELNET, FTP, and MTP. An initial version of TELNET has been working since early in the quarter and has been used to connect successfully to many TCP sites on the ARPANET. Currently both user and server TELNET are available on the VAX. A remaining task is to repackage TELNET so that it can be easily transported to sites outside BBN.

Development and debugging continue on user/server FTP and on MTP. It is expected that working versions of these protocols will be available early in the coming quarter.

9.3 Performance Measurement

Some preliminary performance measurements were made on both the 3BSD and 4.1BSD TCP implementations. The nature of these

measurements was to send data between two processes running on the VAX via a TCP connection that was looped back at the IMP (i.e., out to the IMP and back). The data throughput (not including headers) was measured in each direction. Measurements were made with the default 1K bytes of buffering per send and receive side of the connection, and with 4K bytes of buffering per side, both using 512 byte messages. Typical measurements for the 1K buffering case were 50K bits/sec in each direction. For the 4K case, typical measurements showed 80K bits/sec throughput. With the new 4.1BSD kernel, these figures improved to 70K bits/sec and 120K bits/sec, respectively, a 40-50% performance improvement.

Both sets of measurements were made on stand-alone and moderately loaded systems. Variations of less than 10% were observed in the performance figures. The most significant impact on overall system performance seemed to be an increase in the amount of context switching taking place during the tests. The context switch rate rose from 20-30 switches per second to 90-100 switches per second.

Note that these measurements reflect data throughput only; each 512 byte message requires an additional 52 bytes of TCP, IP, and ARPANET 1822 headers, or 9.8% more data to be transferred with each message than was measured in the tests. The C/30 IMP that the VAX was connected to for these tests is capable of

clocking data out at 200K bits/sec. Taking into account the approximate 10% overhead required for each message sent, this results in a potential 180K bits/sec upper bound on data throughput.

There are several areas of study for performance improvement in the implementation, to try and close the gap between potential and observed throughput. First, there are system improvements that could be made: reducing the amount of context switching overhead entailed by TCP running as a distinct process in the kernel, and improving access times for heavily used data structures by using better search algorithms. Second, there are tuning improvements such as experimenting with larger buffers in the network memory allocation schemes. Finally, there are improvements in the implementation of the protocol, such as improved acknowledgement and retransmission strategies. We will attempt to examine these areas in the months ahead. More thorough performance measurements are needed to accomplish this. Examples are examination of the effect of message size and buffer size on performance, and more detailed study of how network operation affects the rest of the system.

9.4 Future Work

Testing and debugging are continuing on the current version of the TCP. In addition to the performance analysis and improvement work mentioned above, work will begin in the coming quarter on expanding the IP to handle multiple network interfaces and on fully implementing IP-gateway routing and handling of gateway-host protocol messages. This entails generalizing data structures at the local network interface level to handle connections to multiple networks and network interface device drivers. In addition, the IP level will be expanded to maintain routing information, understand and use gateway-host routing advisories, make routing decisions, and handle source quench messages from the gateways.

DISTRIBUTION
[QTR 21]

ARPA
Director (3 copies)
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209
Attn: Program Manager

R. Kahn
V. Cerf
R. Ohlander
D. Adams

DEFENSE DOCUMENTATION CENTER (12 copies)
Cameron Station
Alexandria, VA 22314

DEFENSE COMMUNICATIONS ENGINEERING CENTER
1850 Wiehle Road
Reston, VA 22090
Attn: Lt. Col. F. Zimmerman

DEPARTMENT OF DEFENSE
9800 Savage Road
Ft. Meade, MD 20755
R. McFarland R17 (2 copies)
M. Tinto S46 (2 copies)

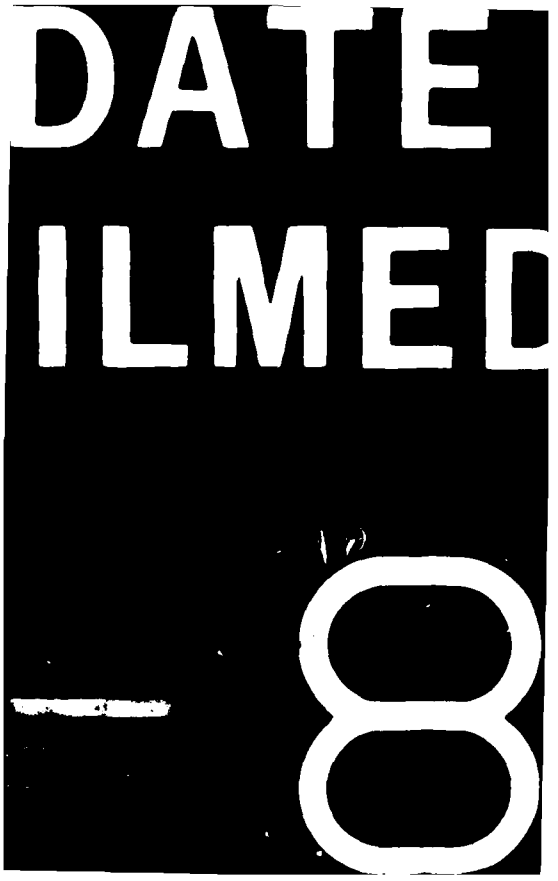
Naval Electronic Systems Command
Department of the Navy
Washington, DC 20360
B. Hughes, Code 6111
F. Deckelman, Code 6131
J. Machado, Code 6134

BOLT BERANEK AND NEWMAN INC.
1701 North Fort Myer Drive
Arlington, VA 22209
E. Wolf

DISTRIBUTION cont'd
[QTR 21]

BOLT BERANEK AND NEWMAN INC.
50 Moulton Street
Cambridge, MA 02138

A. Owen
G. Falk
R. Bressler
A. Lake
J. Robinson
A. McKenzie
F. Heart
P. Santos
R. Brooks
W. Edmond
J. Haverty
D. McNeill
M. Brescia
A. Nemeth
B. Woznick
R. Thomas
W. Milliken
S. Groff
M. Hoffman
R. Rettberg
W. Mann
P. Carvey
D. Hunt
P. Cudhea
L. Evenchik
D. Flood Page
J. Herman
J. Sax
R. Hinden
G. Ruth
S. Kent
R. Gurwitz
E. Starr
Library


DATE
ILMED
8